10/076,537 PTO-892

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/445 — A1

(11) International Publication Number: WO 95/14969

(43) International Publication Date: 1 June 1995 (01.06.95)

(21) International Application Number: PCT/US94/07035

(22) International Filing Date: 21 June 1994 (21.06.94)

(30) Priority Data:
08/159,007 29 November 1993 (29.11.93) US

(71) Applicant: WANG LABORATORIES, INC. [US/US]; One Industrial Avenue, MS 01902B, Lowell, MA 01851 (US).

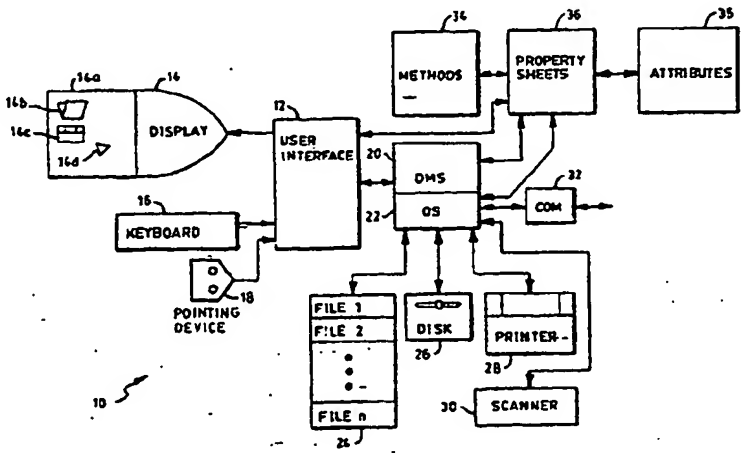
(72) Inventors: BOWMAN, David, M.; 99 Rideout Road, Hollis, NH 03049 (US). GANSZ, Dolerita, J.; 33 Carriage Lane, Stow, MA 01775 (US).

(74) Agent: MILIK, Kenneth, L.; Wang Laboratories, Inc., One Industrial Avenue, MS 01902B, Lowell, MA 01851 (US).

(81) Designated States: AU, CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: METHODS AND APPARATUS FOR CREATING AND MANAGING DYNAMIC PROPERTY SHEETS



(57) Abstract

A method for operating a digital data processing system (10) for modifying the characteristics of an object, such as an application program (applet), a data file, or a device, that is selected from a base class of objects. The method includes a first step-of, in response to an input signal, accessing within a storage device a data structure that is associated with the object. The data structure is referred to herein as a dynamic Property Sheet, and is capable of storing a Property list having at least one entry that is descriptive of at least one property of the base class. The data structure is further capable of storing a Method list having at least one entry that is descriptive of at least one method that is available to the base class. A next step modifies the data structure to add, modify, or delete at least one entry. A next step stores the modified data structure within the storage device so that when the object is subsequently invoked for use, the object is utilized in accordance with the modified data structure.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

METHODS AND APPARATUS FOR CREATING AND MANAGING DYNAMIC PROPERTY SHEETS

FIELD OF THE INVENTION:

This invention relates generally to digital data processors and, in particular, to digital data processors that maintain a record of properties or attributes associated with objects, such as devices and files.

BACKGROUND OF THE INVENTION:

Several known types of digital data processing systems employ a data structure, referred to herein as a Property Sheet, to maintain a record of settings, properties or attributes that are associated with objects. However, in those implementations that are known to the inventors these Property Sheets are static, as opposed to dynamic, structures. That is, once defined the properties of an object remain relatively constant, are not readily changed by a user of the system, and may not be exported for use in defining or modifying the properties of another object of a same, similar, or different class of objects. Known types of conventional implementations also have restrictions on user-visible behavior, and on inconsistencies between device/directory/file objects and nonfile based objects.

OBJECTS OF THE INVENTION:

It is therefore one objective of this invention to provide a dynamic Property Sheet for defining the properties or attributes of an object.

It is another objective of this invention to provide a dynamic Property Sheet having a list of Properties and a list of Methods that are associated with an object.

It is a further objective of this invention to provide a method

of using a dynamic Property Sheet that is associated with a first object to modify the properties of a second object.

It is another objective of this invention to provide a method of using a dynamic Property Sheet that is associated with a first object associated with a first base class or sub-class to modify the properties of a second object associated with a second base class or sub-class.

It is another objective of this invention to provide a method of using a dynamic Property Sheet that is associated with an object associated with a first base class or sub-class to establish an initial set of properties for another, user-defined, base class or sub-class.

It is one further objective of this invention to provide a dynamic Property Sheet that overcomes the deficiencies found in conventional lists of settings for an object, and that enables a user of a data processing system, or an administrator of the system, to add new attributes and/or to redefine and modify object definitions.

SUMMARY OF THE INVENTION

The foregoing and other problems are overcome and the objects of the invention are realized by a method for operating a digital data processing system for modifying the characteristics of an object, such as an application program (applet), a data file, or a device, that is selected from a base class of objects. The method includes a first step of, in response to an input signal, accessing within a storage device a data structure that is associated with the object. The data structure is referred to herein as a dynamic Property Sheet, and is capable of storing a Property list having at least one entry that is descriptive of at least one property of the base class. The data structure is further capable of storing a Method list having at least one entry that is descriptive of at least one method that is

available to the base class. A next step modifies the data structure to add, modify, or delete at least one entry. A next step stores the modified data structure within the storage device so that when the object is subsequently invoked for use, the object is utilized in accordance with the modified data structure.

Each entry of the Property list includes, but is not limited to, at least one Name field that identifies the Property specified by the entry; a Property Value field; a Type field that identifies a type of the Property Value; a field for specifying a Maximum Size of the Property Value; a State field for indicating if the Property list entry is included or removed from the data structure; and a field for specifying a User Interface control function.

Each entry of the Method list includes, but is not limited to, at least one Name field that identifies the Method specified by the entry; a State field for indicating if the Method list entry is included or removed from the data structure; and a Procedure Name field for identifying a program that is executed to accomplish the Method identified by the entry.

The data structure can further include a Name field; a Base Class name field; and a Sub-Class name field.

The method includes a step of merging the data structure with a second data structure associated with the same or another object. This step of merging includes the steps of: for an entry within the Property list of the data structure that does not have a corresponding entry with the same Name field within the Property list of the second data structure, adding the entry to the Property list of the second data structure; and for an entry within the Property list of the data structure that does have a corresponding entry with the same Name field within the Property list of the second data structure, and for which the State field indicates that the entry is removed from the Property list of the

data structure, setting the State field of the corresponding entry of the Property list of the second data structure to indicate that the corresponding entry is removed from the Property list of the second data structure.

The step of merging the data structure with a second data structure associated with the same or another object can also include the steps of: for an entry within the Method list of the data structure that does not have a corresponding entry with the same Name field within the Method list of the second data structure, adding the entry to the Method list of the second data structure; and for an entry within the Method list of the data structure that does have a corresponding entry with the same Name field within the Method list of the second data structure, and for which the State field indicates that the entry is removed from the Method list of the data structure, setting the State field of the corresponding entry of the Method list of the second data structure to indicate that the corresponding entry is removed from the Method list of the second data structure.

The method further includes the steps of (a) copying the data structure as a second data structure; and (b) modifying the Base Class name field to establish a new Base Class having characteristics specified by the second data structure, and/or modifying the Base Sub-Class name field to establish a new Base Sub-Class having characteristics specified by the second data structure.

The methods of the invention are available to an administrator of the system, and selectively to a user of the system. Access top the methods of the invention can be by a user interface having a graphical display screen and a pointing device, or by commands and programs inputted from, by example, a network administrator.

BRIEF DESCRIPTION OF THE DRAWINGS

The above set forth and other features of the invention are made

more apparent in the ensuing Detailed Description of the Invention when read in conjunction with the attached Drawings, wherein:

FIG. 1 is a block diagram of a data processing system that is constructed and operated in accordance with this invention;

FIGs 2A-2C illustrate an inheritance hierarchy that depicts the classes and sub-classes associated with a Class of Object (CCopObject or CCopObj), and which shows the relative position of the class of Property Sheet within the hierarchy;

FIG. 3 depicts an exemplary sequence of user interactions with a dynamic Property Sheet; and

FIG. 4 depicts an exemplary sequence of system administrator interactions with a dynamic Property Sheet.

DETAILED DESCRIPTION OF THE INVENTION

Reference is made to FIG. 1 which illustrates a data processing system 10 that is constructed and operated in accordance with an embodiment of this invention. In this embodiment, a user of the system 10 is enabled to access and interact with a dynamic Property Sheet through the use of a graphical display device, a pointing device, such as a mouse, and a standard keyboard. However, and as will described below, the teaching of the invention applies as well to systems that do not provide a graphical user interface or environment.

More particularly, a user interface 12 is coupled to a graphical display 14, a keyboard 16, and a pointing device 18 for displaying information to, and receiving input from, a user of the system. The display 14 has a display screen 14a on which are displayed text and also graphical images, referred to herein as icons 14b and 14c, and a cursor 14d. The pointing device 18 may be a mouse, trackball, pen, or any suitable real or emulated

device that is capable of providing position data, referenced to the display screen 14a, to the user interface 12. Bidirectionally connected to the user interface 12 is a document management system (DMS) 20 that works in cooperation with an operating system (OS) 22. The DMS 20 is also referred to herein by the acronym "Cop".

The system 10 includes a plurality of data structures and devices, also referred to herein as objects, which may include, but are not limited to, files (FILE_1-FILE_n) stored within a file storage memory 24, a disk 26, a printer 28, a document scanner 30, and a communications port (COM) 32. The OS 22 includes, in a conventional manner, low-level drivers that are used for (a) controlling the creation, storage, and retrieval of the FILES; and (b) operating the disk 26, printer 28, scanner 30, and COM 32 devices.

In general, an object may be a tangible object, such as a printer, or an intangible object, such as a folder, document, image, part of a data base, a user-defined data structure, etc. For a tangible object, such as a printer, it is understood that the object includes the driver and interface software that is required to operate the physical device.

Bidirectionally coupled to the DMS 20, the user interface 12, and the OS 22 is a block 36 which stores and manages the operation of a plurality of data structures referred to herein as "dynamic Property Sheets". Bidirectionally coupled to the block 36 is a memory 34 which stores a plurality of Methods, and a memory 35 which stores a plurality of Attributes. In accordance with an aspect of this invention, the Property Sheets have dynamic, as opposed to static, characteristics and are user-definable and user-modifiable through, for example, the user interface 12.

In general, a dynamic Property Sheet stores at least one Method that is associated with the object from which it is derived (i.e. a Property Sheet created from a printer device has printer device

properties). Typically, a dynamic Property Sheet will include a list of Attributes and a list of Methods that are associated with an object. Once created, the Attributes and Methods can be changed by the user by, for example, using a mouse pointing device 18 to double click on a displayed instance of the Property Sheet. User defined attributes may be added if the class with which the Property Sheet is associated supports user defined attributes. The Methods of the dynamic Property Sheet are stored within the memory 34, and are those Methods that are available for the source object. For example, all printer Methods on an object source instance are a component of the printer Property Sheet method definitions. The user may add, delete, or modify a Method with a user-defined script, an executable program, or a registry method. Using the pointing device 18, in conjunction with the icons displayed on the display screen 14a, a Property Sheet can be "dropped" onto an instance of a 'like' object, (i.e. printer, if derived from a printer) to set the properties of that object for future use, or dropped on a data object, such as a document, thereby associating a set of properties relating to printing, faxing, etc. with the document. These properties are subsequently used when the data object is dropped onto the printer, fax, etc. As such, it may be useful to consider a dynamic Property Sheet as a set of object properties (attributes) and methods for which one may prompt a user.
The Attributes of the dynamic Property Sheet are stored within the memory 35.

In general, a class definition for an object can be obtained, and a dynamic Property Sheet object can be constructed from the class definition. By modifying the object class name, the dynamic Property Sheet can then be used to form a class definition for a new object.

A dynamic Property Sheet can be dropped on an object of 'like type' (i.e., a 'document' property sheet dropped on a 'document'). This has the same effect as selectively changing the attributes and methods associated with the object. This is

accomplished by the block 36 selectively filtering the attributes and methods, as described below.

A dynamic Property Sheet can also be dropped on an object of 'unlike type' (i.e., a 'printer' property sheet dropped on a 'document'). In this case the attributes and methods of the document are not affected. However, the printer dynamic Property Sheet is subsequently used to drive the functionality of a like object. For example, an instance of the 'document' is eventually dropped onto a printer for printing. The attributes and methods associated with the printer dynamic Property Sheet are then used to overide the printer object default settings.

In general, one dynamic Property Sheet of given class can be attached to an object. For example, one document dynamic Property Sheet can be attached to a document object, and one printer dynamic Property Sheet can be attached to a printer object.

As an example of attributes and methods, and considering the COM 32, the object properties (attributes) can include the baud rate, the number of stop bits, and the parity (odd or even), while the object methods can include a required communications protocol, as specified by a user-supplied script, such as a C++ program module.

In accordance with this invention, and as will be described in further detail below, the user or the administrator of the system 10 has an ability to dynamically modify the Property Sheets (hence the term "dynamic Property Sheet"). This overcomes the limitations of known types of conventional approaches which, in general, enable a user to only change a content of a list of properties or settings, but not add or delete properties or settings. For example, one known conventional approach enables a user to change a title associated with a list of attributes or settings, or to change the characteristics of predetermined settings, but not to delete settings from, or add settings to, the list associated with a given object.

Reference is made to FIGs. 2A-2C which illustrate a hierarchy of classes that can be associated with a single Class of Cop Object (CCopObj). The arrow connecting the Classes indicates that the lower level Class inherits from the higher level Class. CCopObj is a base class from which any of the aforementioned objects described above in FIG. 1 may be derived, and can also include, by example, image and text documents, a folder, Optical Character Recognition (OCR) results obtained from the scanner 30, a catalog of documents, and user-defined objects.

A next lower level is the Class Cop (or DMS) FILE (CCopFile) which in turn can be associated with a number of sub-classes, including the Class Cop Property Sheet (CCopPropertySheet) which is of particular interest to the description of this invention. The other sub-classes of CCopFile are shown in FIG. 2A for completeness, and are not germane to an understanding of this invention.

The CCopPropertySheet has four sub-classes associated therewith: User Defined; CCopDevicePropertySheet; CCopAppletPropertySheet (FIG. 2B); and CCopDataPropertySheet (FIG. 2C). As employed herein, an "Applet" is considered to be a small application program that is employed by the Cop DMS 20 during its operation. Each of these sub-classes can give rise to a number of defined sub-classes, as is illustrated in FIGS. 2A-2C. For example, the CCopDevicePropertySheet has the sub-classes of User Defined, Printer, Fax, Scanner, and OCR. The Property Sheet for each of these sub-classes includes the Properties and Methods that define the operation of the device. For example, the Printer Property Sheet can include a list all user-settable attributes of the specific printer (number of copies, landscape or portrait, resolution (draft vs. high resolution), etc; and a list of system or user-supplied Methods, for example, a sequence of control codes used to place the printer into a known state.

In general, the dynamic Property Sheet of this invention has the

following characteristics:

(a) support of user definable classes and sub-classes (principally in a desktop and a server environment), including the addition and subtraction of Methods and Properties (Attributes) from an existing object class/sub-class;

(b) an ability to alter a user-visible presentation of properties and methods so as to enhance the useability and utility of the objects themselves; and

(c) an ability to create intelligent objects (Smart Objects™), such as folders, documents, forms, etc., that are capable of autonomously performing work processes from simple operations, such as store, mail, or fax the objects themselves, routing within a group, routing across multiple system environments, "trash" that knows how and when to empty itself, person/group objects that know the 'rules' for processing work requests and documents or forms, folders that know when their contents are complete and how to subsequently process themselves, and any user-defined process.

The dynamic Property Sheet of the invention further provides:

(d) an ability for users to define and store, either graphically or programically, their own object classes/methods with little or no programming;

(e) a framework for users to develop scripts, programs, dynamic subroutine libraries, object registry methods, or other procedural logic to add Methods to an object class; and

(f) support for dynamically defined object properties at both the class and object instance level.

The dynamic Property Sheet of the invention further provides:

(g) an ability to function as a means of proceduralizing object level operations by parameterization through a Property Sheet (i.e. proceduralizing operations such as setting printer defaults through an object Property Sheet);

(h) a consistent user interface for Property Sheets (dynamic or static) for both devices, directories, and file and non-file based objects (stored in an object repository DMS); and

(i) a method of dynamically defining object classes by means of an extended user interface drag and drop metaphor (drag in conjunction with a user specified character sequence).

In the system 10 applets and workspace objects have both static base properties and an ability to support dynamic Property Sheets. All dynamic Property Sheets are derived from base classes, as depicted in FIGs. 2A-2c. A MakePropertySheet Method, described below, is supported by all of the base classes, by Cop (Document Management) classes, and by other device and administrative classes. A dynamic Property Sheet may not be derived from another dynamic Property Sheet (although it may be copied from another dynamic Property Sheet), but can be derived from an object that already has a Property Sheet of 'like' class associated therewith. A dynamic Property Sheet may be derived from an object which already has a dynamic Property Sheet associated therewith, and which then alters the base class definition.

For example, an instance of a folder class may not have a Property Sheet, but instead has a set of base class properties, some of which may already be initialized. A folder dynamic Property Sheet is created from the object instance (or from the folder class definition itself if a system administrator allows visibility to the storage of class definitions), and the visibility and user interface attributes, contents, and presentation of the object's properties (attributes and methods) is altered to fit a user's specific requirements. The dynamic

Property Sheet is then dropped back onto the folder instance, or otherwise associated therewith, causing the DMS 20 to alter the folder's base properties by "filtering" same and overriding (replacing) those base properties that are specified by the dynamic Property Sheet. As a result, the user may subsequently observe that the base properties of the folder instance have different contents, presentation, and availability of methods.

That is, the folder object instance now has a Property Sheet associated with it which effectively alters the user's view of the class definition. Furthermore, by changing the sub-class name attribute (described below) to be different from what it was before, e.g. 'Folder' to 'Insurance Claim Folder', the user defines a new Cop sub-class within the hierarchy depicted in FIGs. 2A-2C. If the user then subsequently creates a dynamic Property Sheet from the 'Insurance Claim Folder' object, the new dynamic Property sheet includes the properties of the source Property Sheet merged with those that may be part of the object's base Property set. If a copy of the object is made, the contents of the associated Property Sheet is copied with the new object instance.

Storage of dynamic Property Sheets is as the 'Property Sheet' subclass of 'File Objects', with an entry in the object repository (Cop DMS 20 or other DMS if appropriate). The Property Sheet object content is stored as a serialized file. All floating and DMS-stored objects support Property Sheets as a subclass of 'Attachments' to the object. Multiple Property Sheets may be associated with a single object. Property Sheets may be deleted from an object in the same way any other attachment may be deleted.

Property Sheets may be dropped on address book entries and on administrative directory and administrative entries. For example, user or group administrative directory entries may have Property Sheets associated with them which specify the user defaults.

Each dynamic Property Sheet is embodied as a data structure that is stored within the block 36 of FIG. 1. The Property Sheet data structure is comprised of a plurality of fields, in a presently preferred embodiment of this invention, as follows.

Class Definition: DynamicPropertySheet

DATA FIELD	DESCRIPTION
Name	Name of the Property Sheet/blank if untitled
Prop. Sheet Base Class	Name of the base object from which this Property Sheet was derived (i.e. printer folder, etc., initially derived from the source object's base class)
Prop. Sheet User Subclass	User/administrator's Name of the Subclass of this Property Sheet
Property List	List of Entries of a set of properties (attributes) of the source class that this Property Sheet is derived from.
Entry:	Each Entry in the list includes:
Internal name or ID	Identifies the Property or attribute;
External name	Name (keyword) shown to user for Property (i.e. baud rate);
Type	Type of Property Value (unit,

	string, etc.);
Maximum size	If Property is a string, max. length of string, Otherwise, sizeof() attribute;
Value	Property's value;
State	Property's state: included in or removed from the Property List; and
UI Present. Attributes	User Interface Control to use in displaying, Part of Presentation Group, etc., default values.
Method List	List of Entries of a selected set of Methods available on the source class.
Entry:	Each Entry in the Method List includes:
Internal name or ID	Identifies Method;
External name	Name (keyword) shown to user for Method;
Procedure name	Script, executable, or other procedural code (If not present the default is the base class registry entry); and
State	Method's state: included, removed, or additional.

The Property Sheet Base Class field is used when determining if two objects are of 'like type'. For example, a color printer object is a subclass of printer object, which in turn is a subclass of device object. In order for a color printer dynamic Property Sheet to be dropped onto an object associated with a like class, the object of the like class must also have the

Property Sheet Base Class field equal to 'color printer'. That is, a derived class and a parent class must both be equal.

It should be realized that the dynamic Property Sheet of this invention is not limited to only the number and type of data fields that are expressly set forth above. For example, other fields that may be used include a Security attributes field for indicating a class of users, and a Data Base Containment field for indicating appropriate data base table and field mapping information.

The Methods which may appear in the Method List field, in the presently preferred embodiment of the invention, are drawn from those stored within the memory 34 of FIG. 1. System defined Methods are described below. Each definition includes the internal name of the Method, a description of the Method's operation, the arguments associated with the Method, and the Class Functions of the Method.

METHODS:

AddMethod - Adds the method and optional script to the Method List at the position specified.
Method Arguments:
Class Functions:
AddMethod_(method list entry, position)

AddProperty - Adds the Property and optional value to the Property List. The property is not actually (physically) "added", instead its state is changed from removed to included; only the external name and value parts of the entry are changed.
Method Arguments:
Class Functions:
AddProperty_(Property list entry)

DeleteMethod - Physically removes the specified method from the

method list.

Method Arguments:

Class Functions:

DeleteMethod_(method internal name or identifier)

GetBaseClass/SetBaseClass - Gets or sets the base class from which the Property Sheet was derived. By changing this attribute the user can define a new base class.

GetUserSubclass/SetUserSubclass - Get or Set the user defined subclass. The default setting on the UserSubclass is the same as the BaseClass. By changing this attribute the user may define a sub-class.

GetUIEntry/SetUIEntry - Gets or Set the attribute controlling the presentation attributes of an individual Property for the User Interface.

GetMethod

Method Arguments:

Class Functions:

GetMethod_(method list entry) - on input only the internal name or identifier need be set. The complete Method List entry is returned.

GetProperty - Retrieves a copy of the specified Property

Method Arguments:

Class Functions:

GetProperty_(Property list entry) - on input only the internal name or identifier need be set. The complete Property List entry is returned.

ListMethods - Returns the list of methods, option to return all or only active (include state). The returned List is composed of Method List entries, one for each Method in the Method List.

Method Arguments:

Class Functions:

ListMethods_(method list, option = active)

ListProperties - Returns the list of properties, option to return all or only active (included state). The returned List is composed of Property List entries, one for each Property in the Property List.

Method Arguments:

Class Functions:

ListProperties_(property list, option = active)

New -

Method Arguments:

Class Functions:

static New (template = NULL) - UI displayed. The destination and name of the new Property Sheet file is not specified until the file is saved.

static New (dst, name, template = NULL) - UI displayed.

Open -

Method Arguments:

Class Functions:

Open() - UI displayed

RemoveMethod - Removes the method from the Method List. Method is not physically removed, instead the state is set to 'removed'.

Method Arguments:

Class Functions:

RemoveMethod_(method internal name or identifier)

RemoveProperty - Removes the specified Property from its list. The Property is not physically removed, instead the state of the Property is set to 'removed'.

Method Arguments:

Class Functions:

RemoveProperty_(internal name or identifier of

Property)

SetMethod - Sets the specified method according to the information in the entry. Only the external name and script can be replaced with this Method.
Method Arguments:
Class Functions:
SetMethod_(method list entry)

SetProperty - Sets the specified Property according to the information in the entry. Only the external name and value of the entry can be set.
Method Arguments:
Class Functions:
SetSourceProperty_(source class entry)

Serialize - To permanent storage
Method Arguments:
Class Functions:
Serialize_()

View -
Method Arguments:
Class Functions:
View() - User Interface is displayed to the user.

The foregoing Method list is exemplary, in that methods may be deleted from this list or others added.

The list of Methods associated with a dynamic Property Sheet functions as a 'mask' that is employed by the DMS 20 in controlling the presentation of the Methods to the user (for example, via a right mouse click, a menu, or a toolbars). What appears to the user on the display screen 14a would be those Methods found in the registry and base class definition for an object, plus those added by a dynamic Property Sheet (or base class). The dynamic Property Sheet also controls the user

presentation of Methods and Properties. That is, it effectively masks the list of methods/attributes of an object instance or class.

For example, a 'folder' object has open, print, mail, close, properties, and Property_Sheet Methods in its object registry. A dynamic Property Sheet has removed Property_Sheet and mail methods from the list of methods available to the user on the folder instance, and added both check and process methods which are embodied within user-defined scripts or procedural code. The available methods to the end user would thus be open, print, close, properties, check, and process. This reflects a merger of those methods in the registry, the base class, and those associated with the object instance or class from the dynamic Property Sheet. The newly created dynamic Property sheet for the 'Smart Object' folder (having inherent check and process Methods) could then be dropped on the administrative class table (FIGs. 2A-2C) to form the basis for what is effectively a new user-defined class. The system user or administrator could then set the user defined sub-class attribute of the Property Sheet to construct an 'intelligent folder' sub-class having enhanced properties over the 'folder' base class.

As an example, the 'intelligent folder' sub-class includes a user-provided script program that verifies that all required fields of the folder are completed, and that further verifies that no fields contain values that are out of a pre-determined range of values. If the folder contents are verified, then another user-provided script can drop the folder onto the printer 28, for printing in accordance with a dynamic printer-derived Property Sheet. Another user-provided script can also drop the folder onto the COM device 32 for transmission to a centrally-provided network mass storage device.

The dynamic Property Sheets of this invention facilitate other useful operations. For example, a drop of user-defined script on a Property Sheet adds the script as a Method, while a drop of a script on any object also adds the script as a method. That is,

if there is no Property Sheet for the object then a dynamic Property Sheet is generated which includes the dropped script as an Entry within the Method List, in addition to the base class attributes, contents, and base class methods that are associated with the object.

Having described in detail the format of a dynamic Property Sheet, and the Methods that may be included within a dynamic Property Sheet, two examples of the use of the dynamic Property Sheets of this invention are now described with reference to FIGs. 3 and 4.

Turning first to FIG. 3 there is presented an exemplary user interaction with the system 10. Initially, the DMS 20 has cooperated with the user interface 12 to display within the display screen 14a a plurality of icons, such as the folder icon 14b and the printer icon 14c. The user manipulates the pointing device 18 to position the cursor 14d over the printer icon 14c and to select same (Step A). In response, the DMS 20 displays a Printer Applet Popup and the user manipulates the pointing device to select Properties (Step B). In response, the DMS 20 executes the Properties method which displays a dialog box to receive input from the user to alter the printer device properties (attributes) by setting, for example, the default number of copies = 1, resolution = standard, and orientation = portrait (Step C). Assuming for this example that the pointing device 18 is a mouse, the user then performs a Ctl Alt drag operation from the printer applet icon (Step A). In response, the DMS 20 executes the appropriate Methods to create a printer dynamic Property Sheet (Step D). The printer dynamic Property Sheet thus contains copies = 1, resolution = standard, and orientation = portrait. The user then performs a right mouse click (Step E) and selects Open. In response, at Step F newly created Property Sheet is displayed to the user, and the user is enabled to edit same. For example, the user changes orientation = landscape and resolution = high (Step F), and then closes the instance. The modified printer dynamic Property Sheet instance may then be

dropped on the printer applet popup (Step G) to cause the DMS 20 to alter the defaults for that instance to whatever is set in the just modified dynamic Property Sheet. The dynamic Property Sheet may also be dropped onto a data object, such as the folder icon 14b, or an image document to associate a set of properties for printing the object. When a data object having an associated printer-derived dynamic Property Sheet is printed, by invoking the print methods or by a drop operation on the printer applet, the printer 28 uses the folder's printer-derived dynamic Property Sheet to set the appropriate device class properties.

As was described above, new instances of Property Sheet objects can be created by using the drag conventions from the workspace. Property Sheets are a defined subclass of Third Party File Objects and are stored in the Cop DMS 20 hierarchy (FIGs. 2A-2C).

Reference is now made to FIG. 4 for an exemplary system-administrator interaction with the DMS 20. The administrator employs dynamic Property Sheets to define new classes and sub-classes, and to construct useful user objects which relate to the enterprise work functions. Dynamic Property Sheets are the base mechanism for storing class definitions.

As in the example of FIG. 3, the DMS 20 has cooperated with the user interface 12 to display within the display screen 14a the folder icon 14b (referred to here as a 'Claims Folder' icon and the printer icon 14c. The administrator manipulates the pointing device 18 to position the cursor 14d over the claims folder icon 14b and to select same (Step A). In response, the DMS 20 displays a Claims Folder Popup, and the administrator manipulates the pointing device to select Properties (Step B). The administrator then edits the properties of the claims folder object instance. In this example the administrator is prevented from modifying the methods or text labels of the Claims Folder base class.

Assuming also for this example that the pointing device 18 is a mouse, the administrator then performs a Ctl Alt drag operation

from the Folder icon 14b (Step D) to cause the DMS 20 to create a dynamic Property Sheet having all of the now modified properties, and also the base class methods, of the Claims Folder base class. The administrator then performs a single mouse click to bring up the Property Sheet Popup, and a double mouse click to Open the dynamic Property Sheet (Step E). At Step F the administrator adds a Check Completeness Script and a Process Claim Script to the Methods List. The administrator can also add, delete or modify other entries of the Methods List, and can add, delete or modify entries of the Properties List. The modified dynamic Property Sheet can be dropped back onto the Claims Folder icon 14b, thereby causing the DMS 20 to subsequently merge the base properties and methods of the Claims Folder with the properties and methods of the dynamic Property Sheet, as described previously. In this case, for a subsequent instance of the Claims Folder, the newly added script methods Check Completeness and Process Claim would appear to the user to also be a part of the Claim Folder definition (Step G).

The administrator is also enabled to modify the base class or sub-class titles of the dynamic Property Sheet to create a new class or sub-class of the Folder object (for example, 'Self-Checking, Self-Processing Claims Folder' for subsequent selection by the user. Modification of the UI fields also enables a different icon to be displayed for the new object, and enables the new object to have user interface characteristics that differ from the user interface characteristics of the 'Claims Folder' object.

Although the teaching of this invention has been described above primarily in the context of access through the user interface 14, it should be appreciated that the creation and use of the dynamic Property Sheets of this invention can be achieved by a network administrator without the use of the display screen 14a and the pointing device 18. That is, the Property Sheet block 36 can be accessed with a series of instructions from a network (for example, through the COM device 32) to create, modify and revise

the Dynamic Property Sheets, and to create new classes and sub-classes of objects, in accordance with the Methods described above.

It should also be realized that the specific fields of the dynamic Property Sheet, and the specific Methods and arguments thereof, that are described above are not to be read in a limiting sense upon the practice of this invention. That is, other Property Sheet data fields and other or revised Methods may be employed without departing from the scope of the teaching of this invention. In like manner, the system 10 of FIG. 1 may have a number of different physical embodiments, other than that specifically illustrated and described. Furthermore, any software event can be used in place of the mouse-specific events described above to initiate and control the dynamic Property Sheet processing. For example, predetermined keys on the keyboard 16 can be used in place of the mouse 18 to position the cursor 14d and provide input to the user interface 12.

Thus, while this invention has been particularly shown and described with respect to a preferred embodiment thereof, it will be understood by those skilled in the art that changes in form and details may be made therein without departing from the scope and spirit of the invention.

CLAIMS

What is claimed is:

1. A method for operating a digital data processing system for modifying the characteristics of an object that is selected from a base class of objects, comprising the steps of:

in response to an input signal, accessing within a storage means a data structure that is associated with the object, the data structure being capable of storing a first list having at least one entry that is descriptive of at least one property of the base class, the data structure further being capable of storing a second list having at least one entry that is descriptive of at least one method that is available to the base class;

modifying the first list to add, modify, or delete at least one entry;

storing the data structure having the modified first list within the storage means; and

when the object is subsequently invoked for use, utilizing the object in accordance with the modified first list.

2. A method for operating a digital data processing system for modifying the characteristics of an object that is selected from a base class of objects, comprising the steps of:

in response to an input signal, accessing within a storage means a data structure that is associated with the object, the data structure being capable of storing a first list having at least one entry that is descriptive of at least one property of the base class, the data structure further being capable of storing a second list having at least one entry that is descriptive of at least one method that is

available to the base class;

modifying the second list to add, modify, or delete at least one entry;

storing the data structure having the modified second list within the storage means; and

when the object is subsequently invoked for use, utilizing the object in accordance with the modified second list.

3. A method for operating a digital data processing system for modifying the characteristics of an object that is selected from a base class of objects, comprising the steps of:

in response to an input signal, accessing within a storage means a data structure that is associated with a first object, the data structure being capable of storing a first list having at least one entry that is descriptive of at least one property of the base class, the data structure further being capable of storing a second list having at least one entry that is descriptive of at least one method that is available to the base class;

modifying a data structure that is associated with a second object in accordance with the data structure that is associated with the first object so as to replace any entry within the first list of the data structure of the second object with a corresponding entry from the first list of the data structure of the first object;

storing the modified data structure within the memory means; and

when the second object is subsequently invoked for action, utilizing the second object in accordance with the first list of the modified data structure.

4. A method as set forth in claim 3 wherein the step of modifying includes an initial step of selectively revising at least one of the first list and the second list of the data structure of the first object to add, modify, or delete at least one entry.

5. A method for operating a digital data processing system for modifying the characteristics of an object that is selected from a base class of objects, comprising the steps of:

in response to an input signal, accessing within a storage means a first data structure that is associated with a base class of an object, the first data structure being capable of storing a first list having at least one entry that is descriptive of at least one property of the base class, the first data structure further being capable of storing a second list having at least one entry that is descriptive of at least one method that is available to the base class;

copying the first data structure to generate a second data structure;

revising the second list of the second data structure to add or delete at least one entry;

storing the second data structure within the memory means in association with the first data structure; and

when the object is subsequently invoked for action, utilizing the object in accordance with a merged data structure that represents a combination of the first data structure with the second data structure so as to add to the second list of the first data structure any entry that was added to the second list of the second data structure, and to delete from the second list of the first data structure any entry that was deleted from the second list of the second data structure.

6. A method as set forth in claim 5 and further including a step of copying the merged data structure; and a step of establishing a new base class of objects having characteristics specified by the copy of the merged data structure.

7. A method for operating a digital data processing system for establishing the characteristics of objects that are associated with a base class of objects, comprising the steps of:

in response to an input signal, accessing within a storage means a first data structure that is associated with a base class of an object, the first data structure being capable of storing a first list having at least one entry that is descriptive of at least one property of the base class, the first data structure further being capable of storing a second list having at least one entry that is descriptive of at least one method that is available to the base class;

copying the first data structure to generate a second data structure;

revising the second data structure to add, delete, or modify at least one entry;

merging the second data structure with the first data structure so as to add to the first data structure any entry that was added to the second data structure, and to delete from the first data structure any entry that was deleted from the second data structure;

copying the merged data structure; and

establishing a new base class of objects having characteristics specified by the copy of the merged data structure.

8. A method for operating a digital data processing system

for modifying the characteristics of an object that is selected from a base class of objects, comprising the steps of:

in response to an input signal, accessing within a storage means a data structure that is associated with the object, the data structure being capable of storing a Property list having at least one entry that is descriptive of at least one property of the base class, the data structure further being capable of storing a Method list having at least one entry that is descriptive of at least one method that is available to the base class;

modifying the data structure to add, modify, or delete at least one entry;

storing the modified data structure within the storage means; and

when the object is subsequently invoked for use, utilizing the object in accordance with the modified data structure; wherein

each entry of the Property list includes,

at least one Name field that identifies the Property specified by the entry; a Property Value field; a Type field that identifies a type of the Property Value; a field for specifying a Maximum Size of the Property Value; a State field for indicating if the Property list entry is included or removed from the data structure; and a field for specifying a User Interface control function; and wherein each entry of the Method list includes,

at least one Name field that identifies the Method specified by the entry; a State field for indicating if the Method list entry is included or removed from the data structure; and a Procedure Name field for identifying a

program that is executed to accomplish the Method identified by the entry.

9. A method as set forth in claim 8 wherein the data structure is further comprised of a Name field; a Base Class name field; and a Sub-Class name field.

10. A method as set forth in claim 8 and further including a step of merging the data structure with a second data structure associated with the same or another object, wherein the step of merging includes the steps of:

for an entry within the Property list of the data structure that does not have a corresponding entry with the same Name field within the Property list of the second data structure, adding the entry to the Property list of the second data structure; and

for an entry within the Property list of the data structure that does have a corresponding entry with the same Name field within the Property list of the second data structure, and for which the State field indicates that the entry is removed from the Property list of the data structure, setting the State field of the corresponding entry of the Property list of the second data structure to indicate that the corresponding entry is removed from the Property list of the second data structure.

11. A method as set forth in claim 8 and further including a step of merging the data structure with a second data structure associated with the same or another object, wherein the step of merging includes the steps of:

for an entry within the Method list of the data structure that does not have a corresponding entry with the same Name field within the Method list of the second data structure, adding the entry to the Method list of the second data

structure; and

for an entry within the Method list of the data structure that does have a corresponding entry with the same Name field within the Method list of the second data structure, and for which the State field indicates that the entry is removed from the Method list of the data structure, setting the State field of the corresponding entry of the Method list of the second data structure to indicate that the corresponding entry is removed from the Method list of the second data structure.

12. A method as set forth in claim 8 wherein the data structure is further comprised of a Base Class name field, and wherein the method further includes the steps of:

copying the data structure as a second data structure; and

modifying the Base Class name field to establish a new Base Class having characteristics specified by the second data structure.

13. A method as set forth in claim 8 wherein the data structure is further comprised of a Base Sub-Class name field, and wherein the method further includes the steps of:

copying the data structure as a second data structure; and

modifying the Base Sub-Class name field to establish a new Base Sub-Class having characteristics specified by the second data structure.

14. A data processing system, comprising:

means for modifying one or more characteristics of an object that is selected from a base class of objects, said modifying means comprising,

means, responsive to an input signal, for accessing within a storage means a data structure that is associated with the object, the data structure being capable of storing a Property list having at least one entry that is descriptive of at least one property of the base class, the data structure further being capable of storing a Method list having at least one entry that is descriptive of at least one method that is available to the base class;

means for modifying the data structure to add, modify, or delete at least one entry;

means storing the modified data structure within the storage means; and

when the object is subsequently invoked for use, means for utilizing the object in accordance with the modified data structure; wherein

each entry of the Property list includes,

at least one Name field that identifies the Property specified by the entry; a Property Value field; a Type field that identifies a type of the Property Value; a field for specifying a Maximum Size of the Property Value; a State field for indicating if the Property list entry is included or removed from the data structure; and a field for specifying a User Interface control function; and wherein each entry of the Method list includes,

at least one Name field that identifies the Method specified by the entry; a State field for indicating if the Method list entry is included or removed from the data structure; and a Procedure Name field for identifying a program that is executed to accomplish the Method identified by the entry.

15. A data processing system as set forth in claim 14 wherein the data structure is further comprised of a Name field; a Base Class name field; and a Sub-Class name field.

16. A data processing system as set forth in claim 14 and further including means for merging the data structure with a second data structure associated with the same or another object, said merging means comprising:

for an entry within the Property list of the data structure that does not have a corresponding entry with the same Name field within the Property list of the second data structure, means for adding the entry to the Property list of the second data structure; and

for an entry within the Property list of the data structure that does have a corresponding entry with the same Name field within the Property list of the second data structure, and for which the State field indicates that the entry is removed from the Property list of the data structure, means for setting the State field of the corresponding entry of the Property list of the second data structure to indicate that the corresponding entry is removed from the Property list of the second data structure.

17. A data processing system as set forth in claim 14 and further including means for merging the data structure with a second data structure associated with the same or another object, said merging means comprising:

for an entry within the Method list of the data structure that does not have a corresponding entry with the same Name field within the Method list of the second data structure, means for adding the entry to the Method list of the second data structure; and

for an entry within the Method list of the data structure that does have a corresponding entry with the same Name field within the Method list of the second data structure, and for which the State field indicates that the entry is removed from the Method list of the data structure, means for setting the State field of the corresponding entry of the Method list of the second data structure to indicate that the corresponding entry is removed from the Method list of the second data structure.

18. A data processing system as set forth in claim 14 wherein the data structure is further comprised of a Base Class name field, and wherein said data processing system further includes:

means for copying the data structure as a second data structure; and

means for modifying the Base Class name field to establish a new Base Class having characteristics specified by the second data structure.

19. A data processing system as set forth in claim 14 wherein the data structure is further comprised of a Base Sub-Class name field, and wherein said data processing system further includes:

means for copying the data structure as a second data structure; and

means for modifying the Base Sub-Class name field to establish a new Base Sub-Class having characteristics specified by the second data structure.

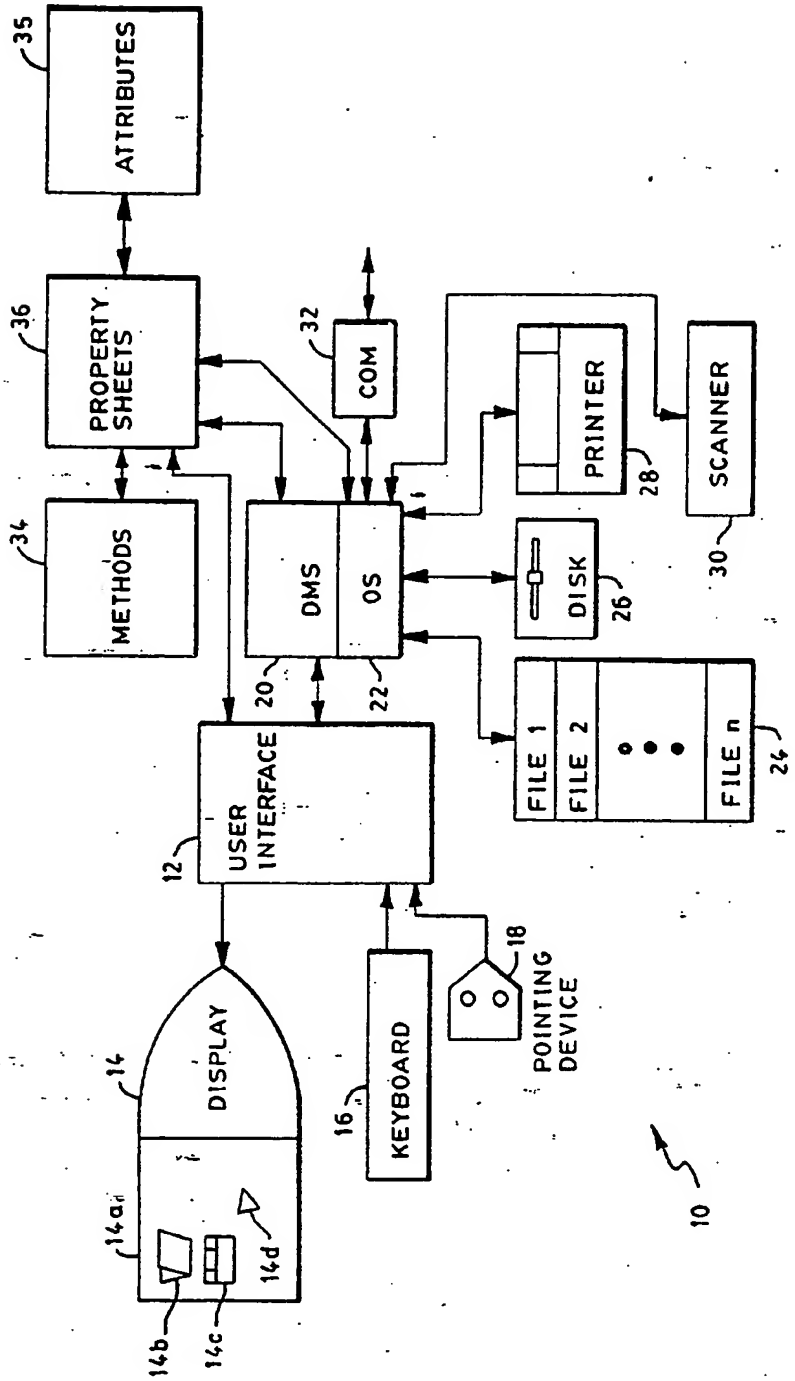

35
ATTRIBUTES
36
PROPERTY SHEETS
34
METHODS
32
COM
DMS
OS
20
22
PRINTER
28
SCANNER
30
DISK
26
FILE 1
FILE 2
FILE n
24
12
USER INTERFACE
14
DISPLAY
14a
14b
14c
14d
16
KEYBOARD
18
POINTING DEVICE
10

FIG. 1

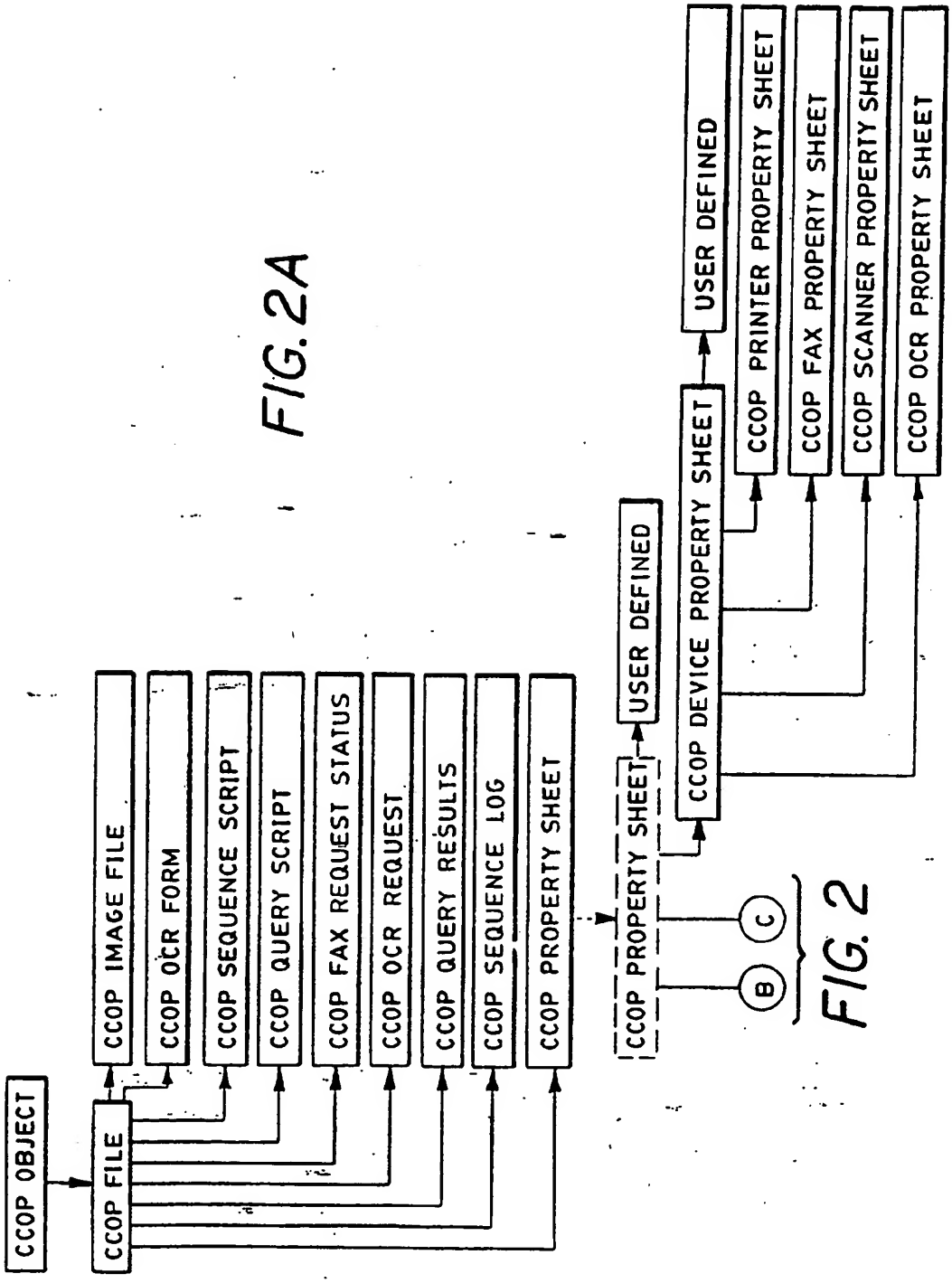

CCOP OBJECT
CCOP FILE
CCOP IMAGE FILE
CCOP OCR FORM
CCOP SEQUENCE SCRIPT
CCOP QUERY SCRIPT
CCOP FAX REQUEST STATUS
CCOP OCR REQUEST
CCOP QUERY RESULTS
CCOP SEQUENCE LOG
CCOP PROPERTY SHEET
CCOP PROPERTY SHEET
USER DEFINED
CCOP DEVICE PROPERTY SHEET
USER DEFINED
CCOP PRINTER PROPERTY SHEET
CCOP FAX PROPERTY SHEET
CCOP SCANNER PROPERTY SHEET
CCOP OCR PROPERTY SHEET
B
C

FIG. 2A

FIG. 2

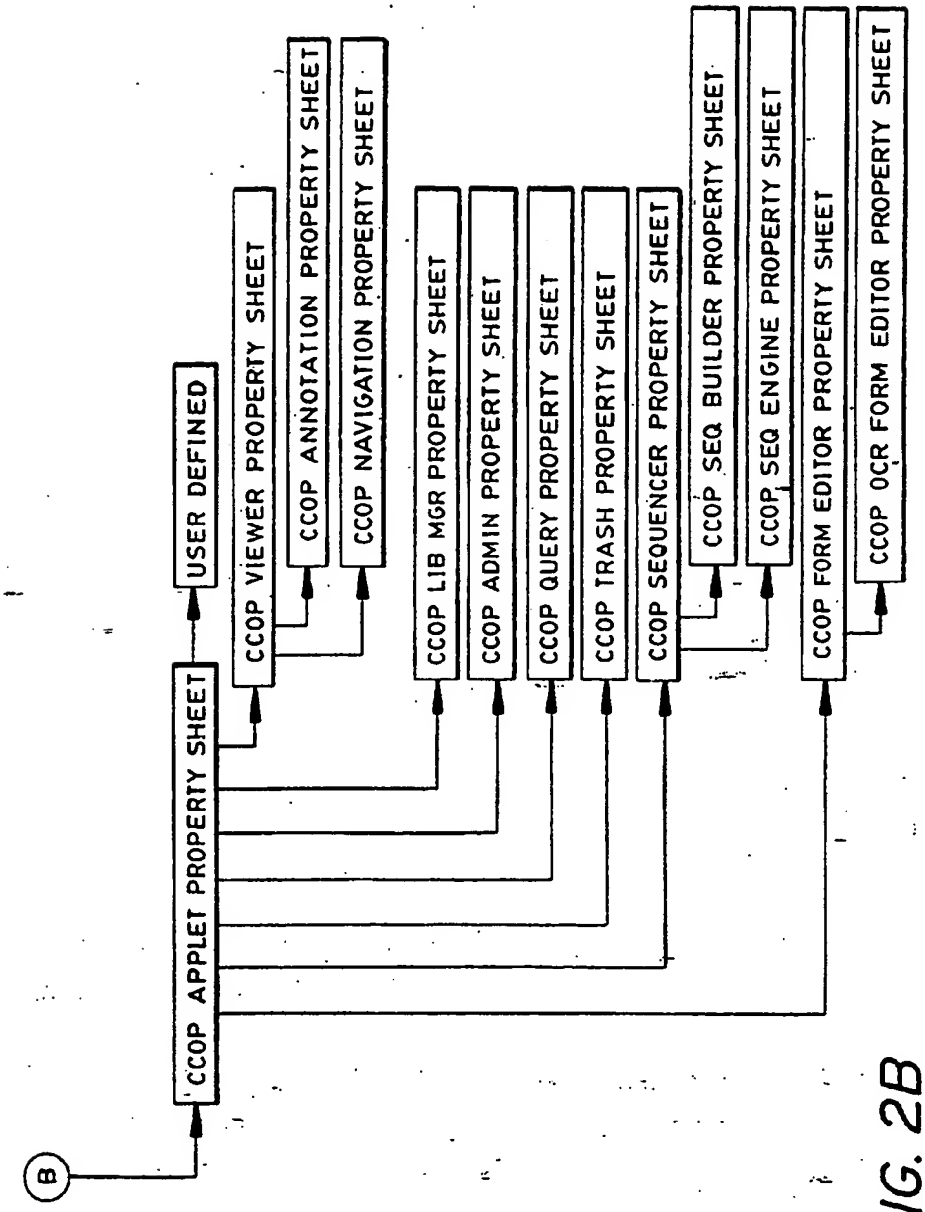

B
CCOP APPLET PROPERTY SHEET
USER DEFINED
CCOP VIEWER PROPERTY SHEET
CCOP ANNOTATION PROPERTY SHEET
CCOP NAVIGATION PROPERTY SHEET
CCOP LIB MGR PROPERTY SHEET
CCOP ADMIN PROPERTY SHEET
CCOP QUERY PROPERTY SHEET
CCOP TRASH PROPERTY SHEET
CCOP SEQUENCER PROPERTY SHEET
CCOP SEQ BUILDER PROPERTY SHEET
CCOP SEQ ENGINE PROPERTY SHEET
CCOP FORM EDITOR PROPERTY SHEET
CCOP OCR FORM EDITOR PROPERTY SHEET

FIG. 2B

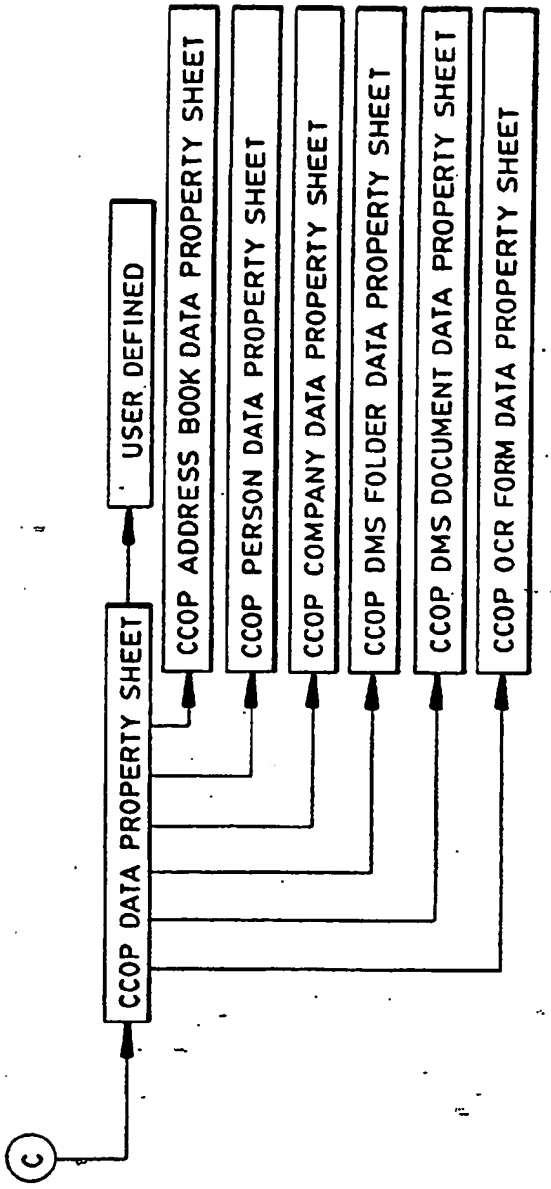

C
CCOP DATA PROPERTY SHEET
USER DEFINED
CCOP ADDRESS BOOK DATA PROPERTY SHEET
CCOP PERSON DATA PROPERTY SHEET
CCOP COMPANY DATA PROPERTY SHEET
CCOP DMS FOLDER DATA PROPERTY SHEET
CCOP DMS DOCUMENT DATA PROPERTY SHEET
CCOP OCR FORM DATA PROPERTY SHEET

FIG. 2C

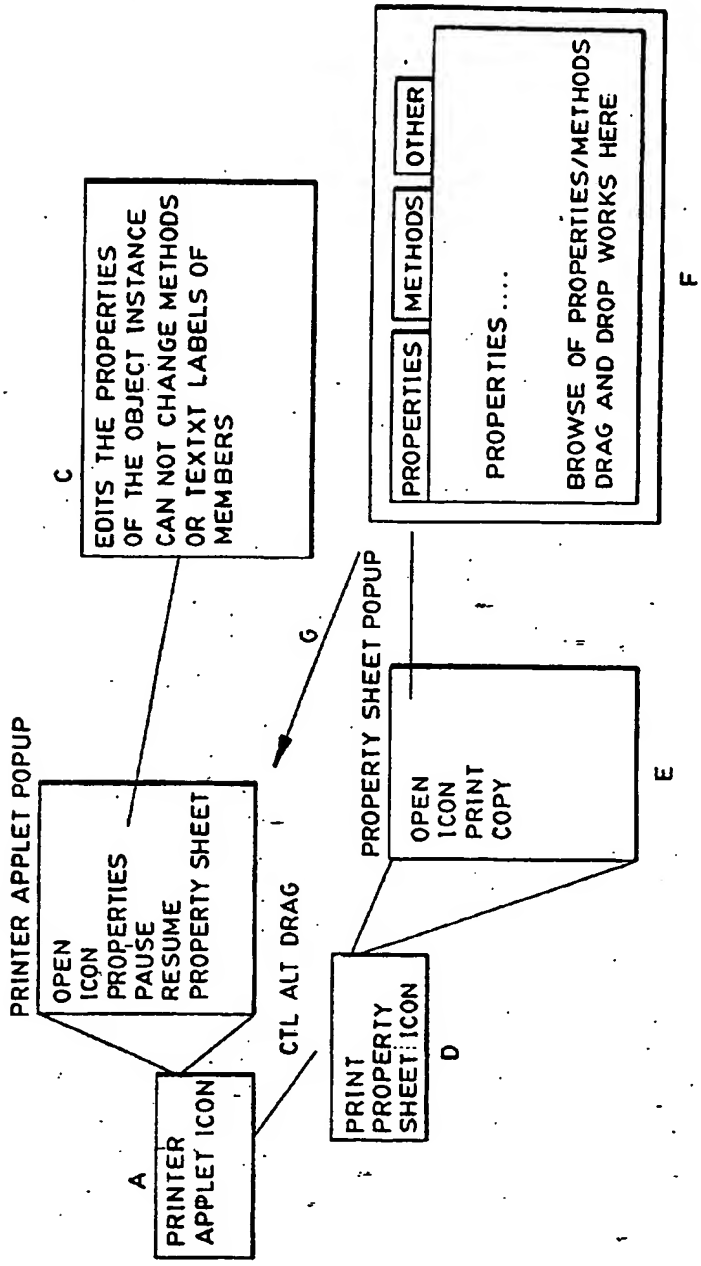

A
PRINTER APPLET ICON
B
PRINTER APPLET POPUP
OPEN ICON
PROPERTIES
PAUSE
RESUME
PROPERTY SHEET
C
EDITS THE PROPERTIES OF THE OBJECT INSTANCE CAN NOT CHANGE METHODS OR TEXTXT LABELS OF MEMBERS
CTL ALT DRAG
D
PRINT PROPERTY SHEET ICON
E
PROPERTY SHEET POPUP
OPEN ICON PRINT COPY
F
PROPERTIES
METHODS
OTHER
PROPERTIES....
BROWSE OF PROPERTIES/METHODS DRAG AND DROP WORKS HERE
G

FIG. 3

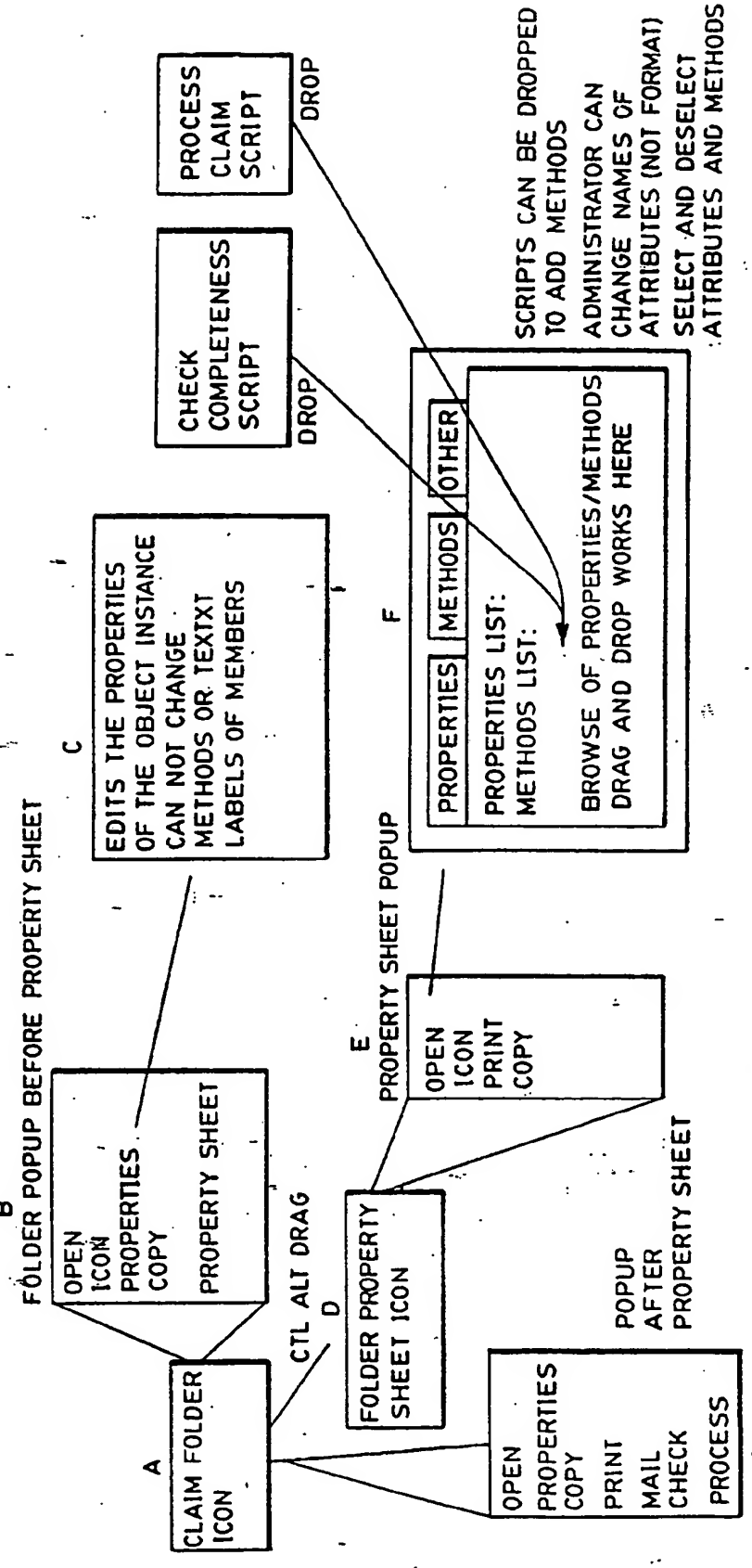

A
CLAIM FOLDER ICON
B
FOLDER POPUP BEFORE PROPERTY SHEET
OPEN
ICON
PROPERTIES
COPY
PROPERTY SHEET
C
EDITS THE PROPERTIES OF THE OBJECT INSTANCE CAN NOT CHANGE METHODS OR TEXTXT LABELS OF MEMBERS
CTL ALT DRAG
D
FOLDER PROPERTY SHEET ICON
E
PROPERTY SHEET POPUP
OPEN
ICON
PRINT
COPY
F
PROPERTIES
METHODS
OTHER
PROPERTIES LIST:
METHODS LIST:
BROWSE OF PROPERTIES/METHODS
DRAG AND DROP WORKS HERE
CHECK COMPLETENESS SCRIPT
DROP
PROCESS CLAIM SCRIPT
DROP
SCRIPTS CAN BE DROPPED TO ADD METHODS
ADMINISTRATOR CAN CHANGE NAMES OF ATTRIBUTES (NOT FORMAT)
SELECT AND DESELECT ATTRIBUTES AND METHODS
POPUP AFTER PROPERTY SHEET
OPEN
PROPERTIES
COPY
PRINT
MAIL
CHECK
PROCESS
G

FIG. 4

INTERNATIONAL SEARCH REPORT

Intern. al Application No
PCT/US 94/07035

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/445

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP,A,0 398 644 (IBM) 22 November 1990 see column 1, line 28 - column 2, line 23 see column 6, line 54 - column 10, line 38 see column 12, line 51 - column 13, line 7 see column 13, line 23 - column 14, line 6 see column 14, line 30 - column 15, line 45	1-19

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 November 1994	01. 12. 94

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk - Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Brandt, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter ial Application No
PCT/US 94/07035

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0398644	22-11-90	CA-A,C	2016396	15-11-90
		JP-A-	3006615	14-01-91